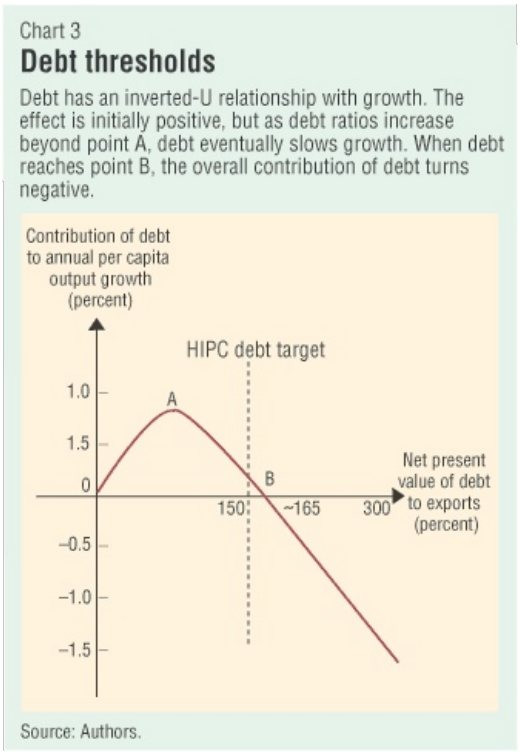
The Influence of External Debt of Economic Growth

Abstract

This project is derived from my graduate thesis on the relationship between external debt and economic growth in Kazakhstan. It adapts an economic growth model based on IMF research, using panel data from 4 countries to assess the impact of external debt on economic growth. This exercise demonstrates data utilization, model building, and curve fitting in Python. However, its results have limited economic significance due to small number of variables and the overall shift in the Kazakh economy towards oil export in the late 2000s, making the model less relevant from the economic standpoint.

Background

Moderate levels of external debt, utilized to fund productive investments, are likely to contribute to economic growth. However, exceeding specific thresholds in indebtedness may impede growth. According to an IMF study, there are two crucial junctures: one where increased debt hampers growth and another where it detrimentally affects growth, ultimately worsening the economic situation of the country.



IMF, Finance & Development, June 2002 - External Debt and Growth

Data source

I am relying on the World Bank Open Data (https://data.worldbank.org/) and their Python API (https://pypi.org/project/wbgapi/).The World Bank Open Data website provides a comprehensive platform for accessing global development data. (https://blogs.worldbank.org/opendata/introducing-wbgapi-new-python-package-accessing-world-bank-data).

Study samples

Study samples represent data collected from 1995 to 2021 in 4 countries (KAZ, KGZ, BLR, RUS).

I collected the following indicators ([code]) for the analysis:

[NY.GDP.MKTP.KD.ZG] GDP growth (annual %)

[DT.TDS.DPPF.XP.ZS] Debt service (PPG and IMF only, % of exports of goods, services and primary income)

[BN.RES.INCL.CD] Reserves and related items (BoP, current US$)

[FP.CPI.TOTL.ZG] Inflation, consumer prices (annual %)

[NE.EXP.GNFS.ZS] Exports of goods and services (% of GDP)

In [93]:
```
!pip install wbgapi
!pip install statsmodels

import wbgapi as wb
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from statsmodels.tsa.statespace.sarimax import SARIMAX
from scipy.optimize import curve_fit
from sklearn.metrics import mean_squared_error, r2_score

import seaborn as sns

from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

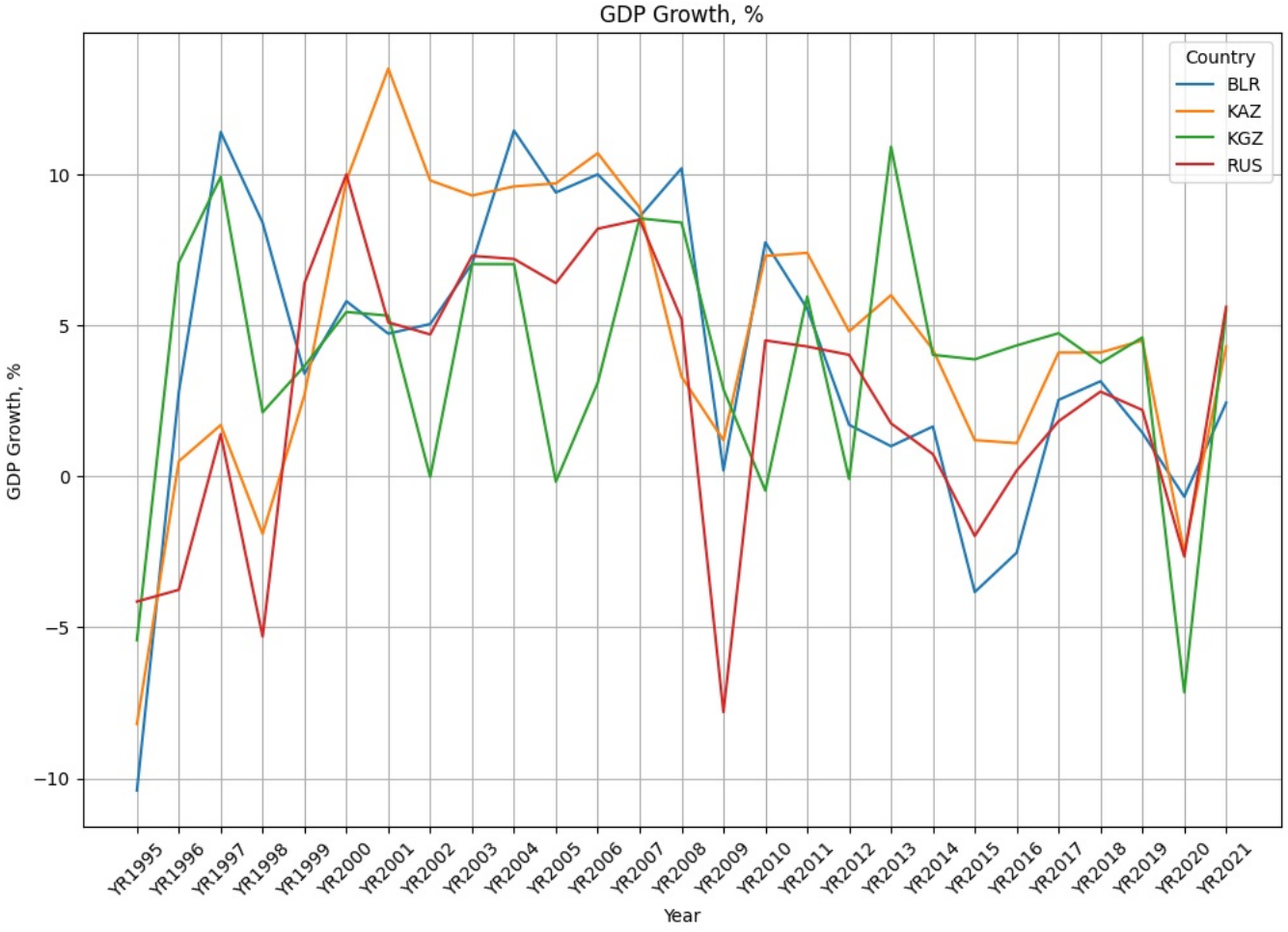
```
Requirement already satisfied: wbgapi in /usr/local/lib/python3.11/dist-packages (1.0.12)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from wbgapi) (2.32.3)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.11/dist-packages (from wbgapi) (6.0.2)
Requirement already satisfied: tabulate in /usr/local/lib/python3.11/dist-packages (from wbgapi) (0.9.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from reques
ts->wbgapi) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->wbgapi)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->wb
gapi) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->wb
gapi) (2024.12.14)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.11/dist-packages (0.14.4)
Requirement already satisfied: numpy<3,>=1.22.3 in /usr/local/lib/python3.11/dist-packages (from statsmodels) (
1.26.4)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in /usr/local/lib/python3.11/dist-packages (from statsmodels)
(1.13.1)
Requirement already satisfied: pandas!=2.1.0,>=1.4 in /usr/local/lib/python3.11/dist-packages (from statsmodels
) (2.2.2)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packages (from statsmodels) (1.0.
1)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from statsmodels) (2
4.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas!=
2.1.0,>=1.4->statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas!=2.1.0,>=1.
4->statsmodels) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas!=2.1.0,>=
1.4->statsmodels) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2
->pandas!=2.1.0,>=1.4->statsmodels) (1.17.0)
```

In [94]:
```
# Fetching data from World Bank(1995-2021)

wb_dataset = wb.data.DataFrame(
    ['NY.GDP.MKTP.KD.ZG', 'DT.TDS.DPPF.XP.ZS', 'BN.RES.INCL.CD', 'FP.CPI.TOTL.ZG', 'NE.EXP.GNFS.ZS'],
    ['KAZ', 'KGZ', 'RUS', 'BLR'], range(1995, 2022), columns='series')

# Rename columns
wb_dataset.rename(
    columns={
        'NY.GDP.MKTP.KD.ZG': 'gdp_growth',
        'DT.TDS.DPPF.XP.ZS': 'debt_service',
        'BN.RES.INCL.CD': 'reserves',
        'FP.CPI.TOTL.ZG': 'inflation',
        'NE.EXP.GNFS.ZS': 'export',
        'time': 'year',
        'economy': 'country'
    },
    inplace=True
)
# wb_dataset has a hierarchical index (MultiIndex) consisting of economy and time instead of those being column
# Rename index levels
wb_dataset.index.names = ['country', 'year']

print(wb_dataset)
```

```
                    reserves  debt_service   inflation     export  gdp_growth
country year
BLR     YR1995 -7.843837e+07      3.359242  709.346032  49.665165  -10.400001
        YR1996 -2.141586e+08      1.604090   52.712077  46.349804    2.800005
        YR1997  6.505104e+07      1.700816   63.937366  59.859881   11.400005
        YR1998 -3.193342e+08      1.921961   72.869717  59.051129    8.399991
        YR1999  1.989385e+07      3.378055  293.678751  59.203341    3.399999
...                      ...           ...         ...        ...         ...
RUS     YR2017  2.262769e+10           NaN    3.683329  26.090881    1.825790
        YR2018  3.819750e+10           NaN    2.878297  30.793257    2.807245
        YR2019  6.648409e+10           NaN    4.470367  28.433431    2.198076
        YR2020 -1.375408e+10           NaN    3.381659  25.522186   -2.653655
        YR2021  6.356691e+10           NaN    6.694459  29.771214    5.614290

[108 rows x 5 columns]
```

In [95]:

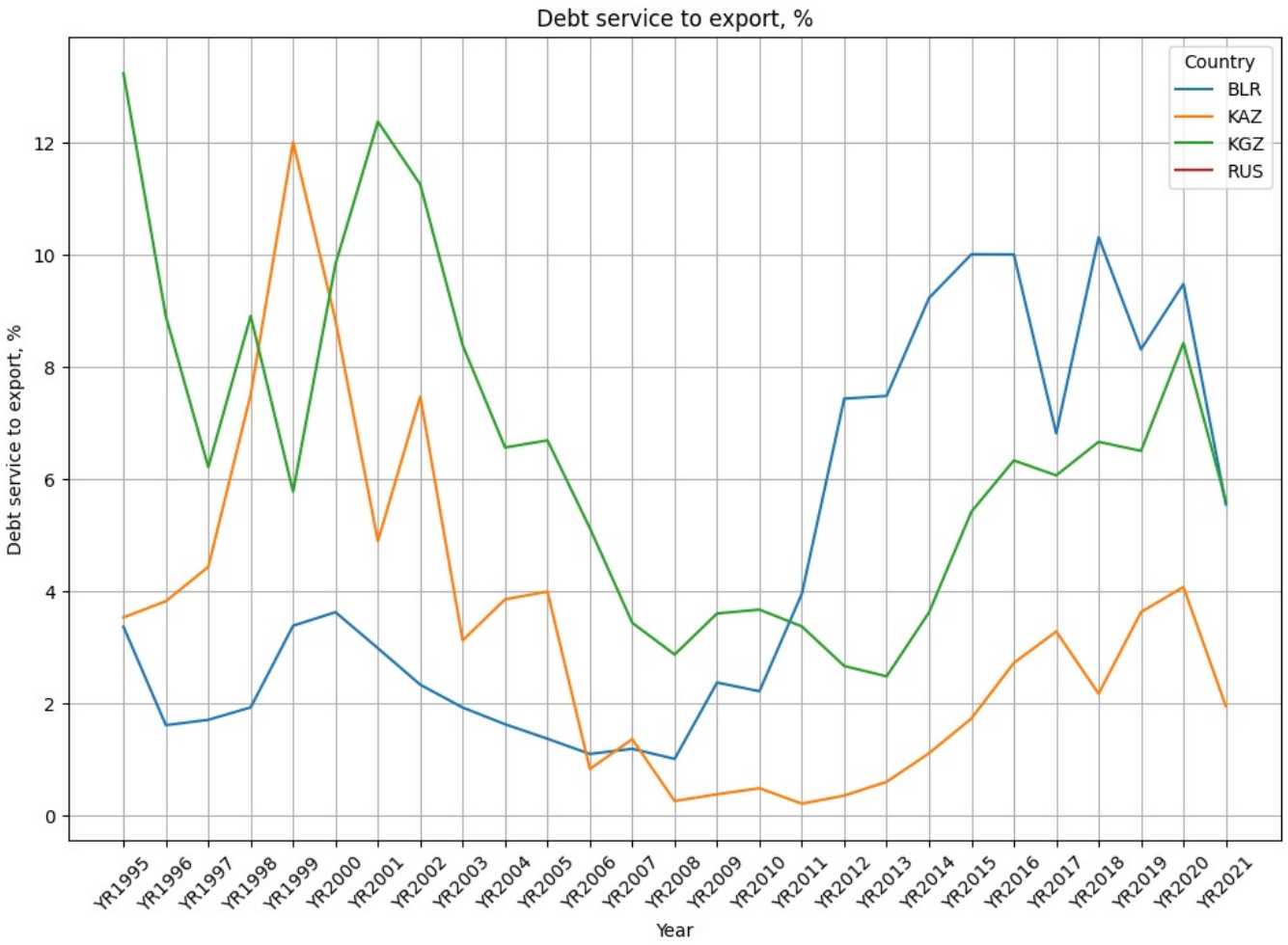
```python
# Vizualizing the GDP growth data

gdp_data = wb_dataset['gdp_growth'].unstack(level='country')
plt.figure(figsize=(12, 8))

for country in gdp_data.columns:
    plt.plot(gdp_data.index, gdp_data[country], label=country)

plt.title("GDP Growth, %")
plt.xlabel("Year")
plt.ylabel("GDP Growth, %")
plt.xticks(rotation=45)
plt.grid(True)
plt.legend(title="Country")

plt.show()
```



In [96]:

```python
# Vizualizing debt service to export data

gdp_data = wb_dataset['debt_service'].unstack(level='country')

plt.figure(figsize=(12, 8))

for country in gdp_data.columns:
    plt.plot(gdp_data.index, gdp_data[country], label=country)

plt.title("Debt service to export, %")
plt.xlabel("Year")
plt.ylabel("Debt service to export, %")
plt.xticks(rotation=45)
plt.grid(True)
plt.legend(title="Country")
```

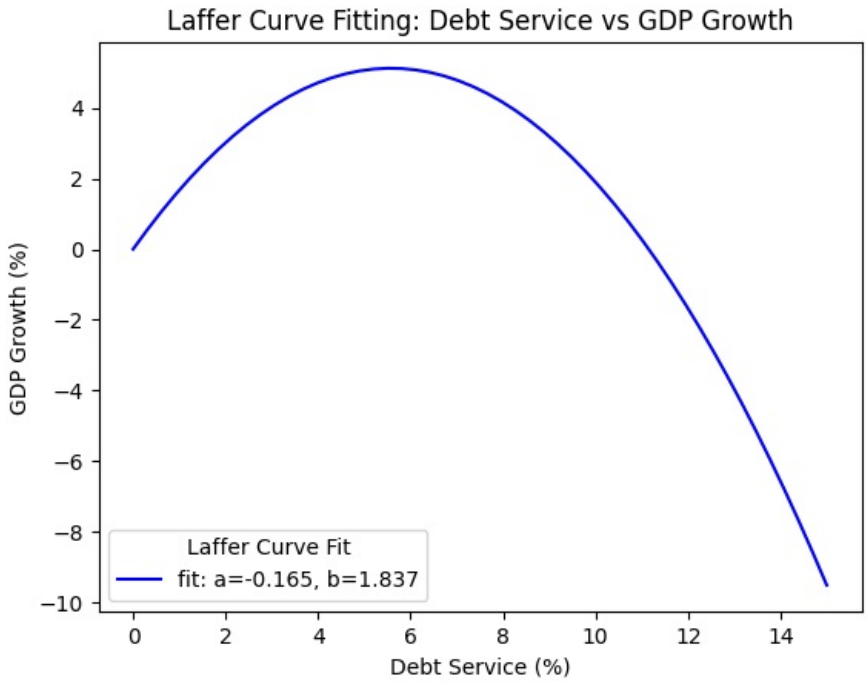
```
plt.show()
```



Curve fitting

We are observing similar relation between debt and growth (IMF, Pattillo et al. [1]) in our dataset:

In [97]:
```
def laffer_curve(x, a, b):
    return a * x ** 2 + b * x

x = wb_dataset['debt_service']
y = wb_dataset['gdp_growth']

# Remove NaN values
valid_data = wb_dataset.dropna(subset=['debt_service', 'gdp_growth'])

x_valid = valid_data['debt_service']
y_valid = valid_data['gdp_growth']

# Curve fitting
popt, pcov = curve_fit(laffer_curve, x_valid, y_valid)

# Generate data points for plotting the fitted curve
xdata = np.linspace(0, 15, 50)
plt.plot(xdata, laffer_curve(xdata, *popt), 'b-', label='fit: a=%5.3f, b=%5.3f' % tuple(popt))


plt.title("Laffer Curve Fitting: Debt Service vs GDP Growth")
plt.xlabel("Debt Service (%)")
plt.ylabel("GDP Growth (%)")
plt.legend(title="Laffer Curve Fit")

plt.show()

# Evaluate the model
f = laffer_curve(x_valid, *popt)
print("Params: ", popt)
print("Condition number of the covariance matrix: ", np.linalg.cond(pcov))
print("Mean Squared Error: ", mean_squared_error(y_valid, f))
print("R2 Score: ", r2_score(y_valid, f))
```




```
Params:  [-0.16473626  1.83697186]
Condition number of the covariance matrix:  785.2127121893989
Mean Squared Error:  25.003315169231033
R2 Score:  -0.17135492277706743
```

ARIMAX modeling and Simulations

I am using Statsmodels, a Python library for statistical models for data analysis.

In [104…

```python
# Data Preparation
wb_dataset = wb_dataset.reset_index()
wb_dataset.set_index(['country', 'year'], inplace=True)

# Define the endogenous and exogenous variables
y = wb_dataset['gdp_growth']
X = wb_dataset[['debt_service', 'inflation', 'export', 'reserves']]

# Remove NaN values
valid_data = pd.concat([y, X], axis=1).dropna()
y_valid = valid_data['gdp_growth']
X_valid = valid_data[['debt_service', 'inflation', 'export', 'reserves']]

# Fit ARIMAX Model
model = SARIMAX(endog=y_valid, exog=X_valid, order=(1, 0, 1), enforce_stationarity=False, enforce_invertibility
results = model.fit()

print(results.summary())

# Simulate future data using Kazakhstan's average
# Calculate average values for Kazakhstan
wb_dataset_kz = wb_dataset.reset_index().query("country == 'KAZ'")

# Exclude non-numeric columns
numeric_cols = ['debt_service', 'inflation', 'export', 'reserves']
kz_means = wb_dataset_kz[numeric_cols].mean().to_frame().T

# Add back non-numeric columns
kz_means['country'] = ['KAZ']
kz_means['year'] = ['FUTURE']
kz_means.set_index(['country', 'year'], inplace=True)

# Repeat for 10 years
future_exog = kz_means.loc[kz_means.index.repeat(10)][['debt_service', 'inflation', 'export', 'reserves']]

# Generate forecasts
forecast = results.get_forecast(steps=10, exog=future_exog)
forecast_mean = forecast.predicted_mean
forecast_ci = forecast.conf_int()
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:             gdp_growth   No. Observations:                   80
Model:               SARIMAX(1, 0, 1)   Log Likelihood                -207.266
Date:                Mon, 20 Jan 2025   AIC                            428.532
Time:                        08:34:14   BIC                            445.029
Sample:                             0   HQIC                           435.136
                                  - 80
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
debt_service    -0.3664   1.99e-19  -1.84e+18      0.000      -0.366      -0.366
inflation       -0.0225   1.45e-18  -1.55e+16      0.000      -0.022      -0.022
export           0.1354   1.18e-18   1.15e+17      0.000       0.135       0.135
reserves      1.695e-10   3.18e-10      0.534      0.594   -4.53e-10    7.92e-10
ar.L1            0.7337   3.67e-21      2e+20      0.000       0.734       0.734
ma.L1           -0.3803    3.4e-20  -1.12e+19      0.000      -0.380      -0.380
sigma2          11.7210   1.06e-20   1.11e+21      0.000      11.721      11.721
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 0.36
Prob(Q):                              0.96   Prob(JB):                         0.84
Heteroskedasticity (H):               1.68   Skew:                            -0.16
Prob(H) (two-sided):                  0.19   Kurtosis:                         2.93
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 4.65e+45. Standard errors may be unst
able.
```

```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported ind
ex was provided. As a result, forecasts cannot be generated. To use the model for forecasting, use one of the s
upported classes of index.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported ind
ex was provided. As a result, forecasts cannot be generated. To use the model for forecasting, use one of the s
upported classes of index.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/base/model.py:607: ConvergenceWarning: Maximum Likelihood o
ptimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index
is available. Prediction results will be given with an integer index beginning at `start`.
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: FutureWarning: No supported inde
x is available. In the next version, calling this method in a model without a supported index will result in an
exception.
  return get_prediction_index(
```

In [106…

```python
# Optimistic scenario: Adjust future debt_service to 6.0
future_exog['debt_service'] = 6.0

# Generate forecasts for the optimistic scenario
forecast_optimistic = results.get_forecast(steps=10, exog=future_exog)
forecast_mean_optimistic = forecast_optimistic.predicted_mean

# Calculate cumulative GDP growth
forecast_df = pd.DataFrame({'gdp_growth': forecast_mean_optimistic})
forecast_df['gdp_growth_accum'] = 1 + forecast_df['gdp_growth'] / 100
forecast_df['gdp_growth_accum'] = forecast_df['gdp_growth_accum'].cumprod()

# Create a DataFrame for scenarios
scenarios = pd.DataFrame()
scenarios['optimistic'] = forecast_df['gdp_growth_accum']
scenarios['year'] = range(len(scenarios))
scenarios['year'] = scenarios['year'] + 2022
```

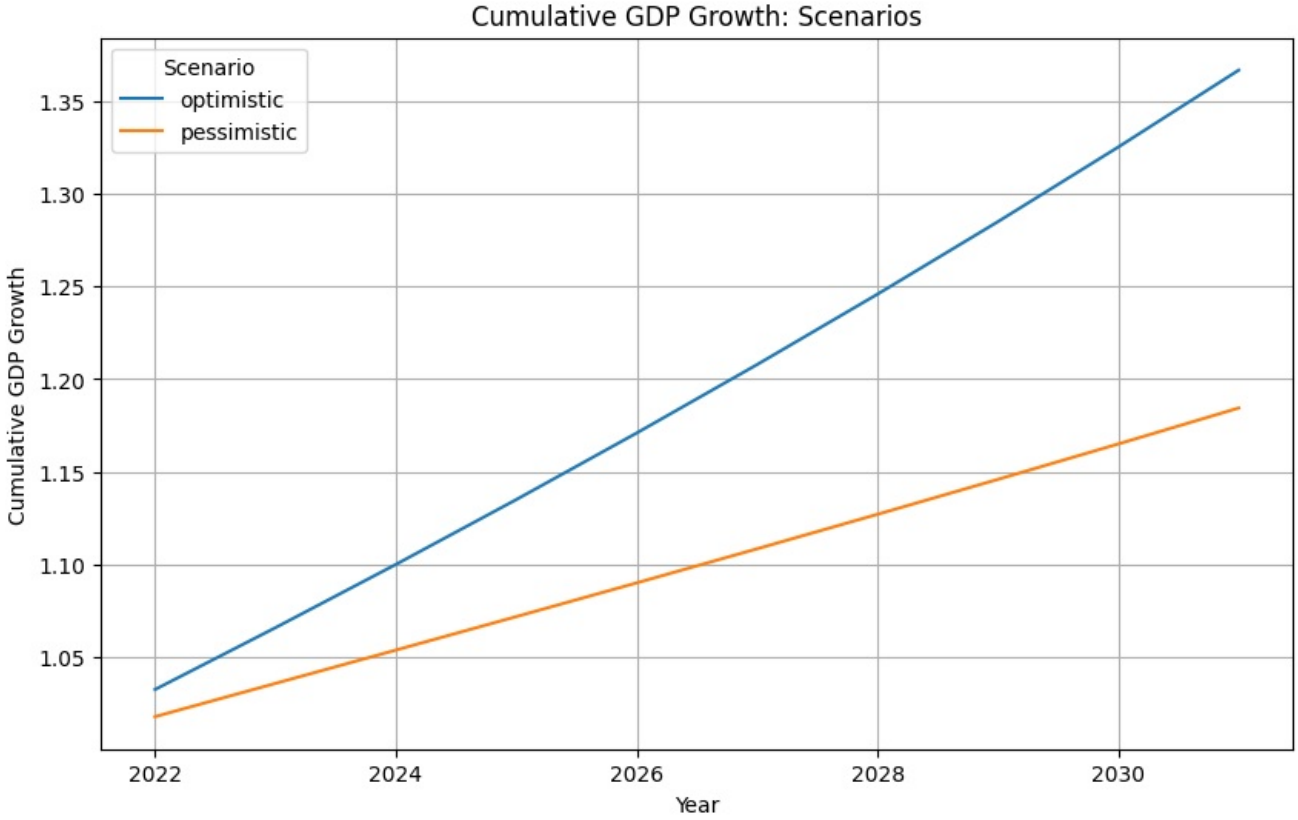
```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index
is available. Prediction results will be given with an integer index beginning at `start`.
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: FutureWarning: No supported inde
x is available. In the next version, calling this method in a model without a supported index will result in an
exception.
  return get_prediction_index(
```

In [107…

```python
# Pessimistic scenario: Adjust future debt_service to 10.0
future_exog['debt_service'] = 10.0

# Generate forecasts for the pessimistic scenario
forecast_pessimistic = results.get_forecast(steps=10, exog=future_exog)
forecast_mean_pessimistic = forecast_pessimistic.predicted_mean

# Calculate cumulative GDP growth for the pessimistic scenario
forecast_df_pessimistic = pd.DataFrame({'gdp_growth': forecast_mean_pessimistic})
forecast_df_pessimistic['gdp_growth_accum'] = 1 + forecast_df_pessimistic['gdp_growth'] / 100
forecast_df_pessimistic['gdp_growth_accum'] = forecast_df_pessimistic['gdp_growth_accum'].cumprod()

# Add pessimistic scenario to the scenarios DataFrame
```

```
scenarios['pessimistic'] = forecast_df_pessimistic['gdp_growth_accum'].values

# Set 'year' as the index
scenarios.set_index(['year'], inplace=True)

# Plot the optimistic and pessimistic scenarios
scenarios.plot.line(title="Cumulative GDP Growth: Scenarios", figsize=(10, 6))
plt.xlabel("Year")
plt.ylabel("Cumulative GDP Growth")
plt.grid(True)
plt.legend(title="Scenario")
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index
is available. Prediction results will be given with an integer index beginning at `start`.
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: FutureWarning: No supported inde
x is available. In the next version, calling this method in a model without a supported index will result in an
exception.
  return get_prediction_index(
```



Conclusion

This work models and forecasts cumulative GDP growth under two scenarios—optimistic and pessimistic—using an ARIMAX model.

Optimistic Scenario: Future debt service is set to 6.0%, predicting lower financial burdens. Cumulative GDP growth is calculated based on forecasted GDP growth rates. Pessimistic Scenario: Future debt service is increased to 10.0%, reflecting higher financial burdens. Forecasted GDP growth rates are used to compute cumulative GDP growth under this scenario. Both scenarios are plotted to visualize their impact on GDP growth over a 10-year horizon, enabling clear comparisons of economic outcomes under varying debt service conditions. The accumulated growth over next 10 years in the optimistic scenario is 18% higher than in the pessimistic scenario.

References

Pattillo, C. A., Poirson, H., & Ricci, L. (2002). External Debt and Growth. IMF Working Paper No. 02/69. International Monetary Fund, Research Department. [Washington, D.C.]. https://www.imf.org/external/pubs/ft/wp/2002/wp0269.pdf

https://www.imf.org/external/pubs/ft/fandd/2002/06/pattillo.htm